In re Application of COLLERAN et al.
Application No. 09/543,930

Amendments to the Claims

1. – 6. (Canceled)

7. (Currently Amended) A method for managing an original user interface for an application, the method comprising:

signaling a hung state for the application;

creating a ghost user interface in response to the hung state that appears in place of the original user interface, wherein creating the ghost user interface includes:

creating a ghost thread for the ghost user interface; and

creating the ghost user interface in the area occupied by the original user interface;

detecting activity in an event queue of the application, wherein detecting activity in the event queue of the application includes:

placing a high priority special event in the event queue; and

detecting a message generated in response to the high priority special event;

replacing the ghost user interface with the original user interface in response to the detection of activity in the event queue of the application; and

~~The method of claim 6 further comprising~~ directing input events corresponding to the area covered by the ghost user interface, which includes at least some of the area covered by the original user interface, to the ghost thread.

8. (Previously Presented) The method of claim 7 further comprising having the ghost thread handle at least one event corresponding to the area covered by the ghost user interface.

9. (Previously Presented) The method of claim 7 further comprising having the ghost thread cache at least one event corresponding to the area covered by the ghost user interface to the ghost thread for handling by the application.

10. (Previously Presented) The method of claim 7 further comprising having the ghost thread handle at least one event corresponding to a minimization operation in the area covered by the ghost user interface.

11. (Previously Presented) The method of claim 7 further comprising having the ghost thread handle at least one event corresponding to a resizing operation in the area covered by the ghost user interface.

12. (Previously Presented) The method of claim 7 further comprising having the ghost thread handle at least one event corresponding to a close operation in the area covered by the ghost user interface.

13. (Previously Presented) The method of claim 7 further comprising having the ghost thread handle at least one event corresponding to a move operation in the area covered by the ghost user interface.

14. (Previously Presented) The method of claim 7 further comprising having the ghost thread cache at least one event corresponding to a keyboard input corresponding to the ghost user interface.

15. (Previously Presented) The method of claim 7 further comprising having the ghost thread handle at least one event corresponding to a keyboard input corresponding to the ghost user interface.

16. (Previously Presented) A method for replacing a user interface element created by a application by a substitute user interface element created by a scheduled code path in a multithreaded computing environment, the method comprising:

detecting a flip-window signal while the user interface element is being displayed;

in response to the flip-window signal, creating the substitute user interface element in the context of the scheduled code path;

superimposing the substitute user interface element over a first-user-interface-element-occupied real estate; and

caching input corresponding to the substitute user interface element wherein the cached input is subsequently handled by the application.

17. (Original) The method of claim 16 wherein the step of creating includes creating the scheduled code path if the scheduled code path does not exist.

18. (Previously Presented) The method of claim 16 further comprising replacing a second user interface element created by a second application by a second substitute user interface element created by the scheduled code path responsively to a second flip-window signal.

19. (Previously Presented) The method of claim 16 further comprising painting over the user interface element responsively to the flip-window signal.

20. (Previously Presented) The method of claim 16 wherein caching input includes input from a plurality of events and/or a plurality of messages.

21. (Previously Presented) The method of claim 16 further comprising caching input corresponding to the application for forwarding to the application in response to the flop-window signal.

22. (Previously Presented) The method of claim 18 further comprising selecting and discarding at least some of the cached input.

23. (Previously Presented) The method of claim 16 wherein caching input includes forwarding the input to the scheduled code path.

24. (Previously Presented) The method of claim 20 further comprising handling at least some of the input by the scheduled code path.

25. (Previously Presented) The method of claim 24 further comprising the scheduled code path handling at least one event in the input by terminating the application.

26. (Previously Presented) The method of claim 16 further comprising placing a special message or event in a queue for the application to generate a flop-window signal.

27. (Original) The method of claim 16 wherein the scheduled code path is a thread.

28. – 30. (Canceled)

31. (Currently Amended) A multithreaded computing system for executing an application having a user interface, wherein if the application is non-responsive to user input, the user interface is replaced by a ghost user interface, the system comprising:

a non-responsive-application detecting code for detecting a non-responsive application;

a ghost thread for creating the ghost user interface to replace the user interface in response to detection of a non-responsive application;

a high priority special entry in a queue for the application for detecting if the application is responsive;

a plurality of computer executable instructions for destroying the ghost user interface in response to handling of the high priority special entry by the application; and

~~The system of claim 30 further comprising~~ a cache of events and messages for handling by the application wherein the events and messages are directed at the ghost thread.

32. – 35. (Canceled)

36. (New) The multithreaded computing system of claim 31, wherein the non-responsive-application detecting code comprises code for detecting a message generated in response to the high priority special entry in the queue for the application.

37. (New) The multithreaded computing system of claim 31, wherein the ghost thread handles at least one event corresponding to an area covered by the user interface.

38. (New) The multithreaded computing system of claim 31, wherein at least one entry in the cache of events and messages is selectively discarded.